关于产生式系统并行执行与收敛性问题的探讨*

郭福顺 宋 震 孟 遥 廖明宏
(哈尔滨工业大学计算机系 哈尔滨 150001)

摘 要 产生式系统引入并行技术后，出现了两个新的困难问题：相容性问题和收敛性问题。基于并行知识库机PKBM95，为了解决收敛性问题，本文给出了一种称为规则顺序锁定的方法。另外，为了发现隐循环以及为了提高性能，我们给出了一系列动态分析和测试的方法，如路径跟踪，等等。

关键词 并行知识库机，收敛性问题，规则顺序锁定

1 引 言

一个一般的产生式系统，尽管其程序结构表面上看起来很简单，但由于其中可能有很难发现的"隐循环"，即便是在串行执行时，出现不收敛情况的可能性已是不得不引起重视的问题。何况，产生式系统的并行执行使收敛性(covergence)[1]问题更加复杂。一个产生式系统的收敛性，即使是在串行的情况下已得到验证，但在并行执行时仍然未必能收敛。因此，在产生式并行执行的知识库系统中，应采取必要的措施，尽可能避免不收敛危机的出现。本文旨在阐述我们在并行知识库机PKBM95[2]中所提出的产生式系统并行划分和并行执行方法，以及应付不收敛危机的基本措施。

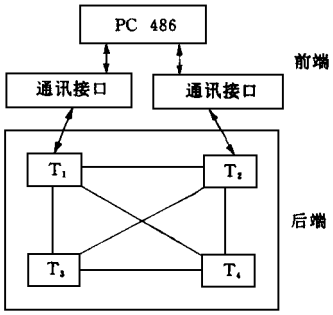


图1 PKBM95硬件结构

PKBM95由前端机和后端机组成，其硬件结构如图1。后端机是前端机(PC486)的一块插板。它以四个Transputer CPU为中心，连成并行处理系统，用于组织Rete网[3]。前端机除了完成系统总控、产生式程序并行划分，对后端机的管理等通常职责之外，在产生式系统并行执行时，还模拟一台Transputer CPU的活动，以使前后端机的负载更接近于平衡。因此，整个硬件系统支持两个层次上的并行推理，一是前后端机之间的并行推理，二是后端机各处理单元之间的并行推理。关于PKBM95的软、硬件设计详情见[4]。本文讨论的并行处理技术、就是在这样的软、硬件环境中实现的。

1996-12-16收稿 * 该项目得到国家自然科学基金资助。郭福顺，教授，研究方向为并行处理、人工智能、操作系统。宋震，博士生，研究方向为人工智能，并行处理。孟遥，硕士生，研究方向为人工智能，并行处理。廖明宏，博士，副教授，研究方向为人工智能，并行处理。

2　产生式规则顺序锁定

让我们首先从一个产生式规则块(作为单一的块)串行执行的角度来分析规则的执行顺序与收敛性的关系。然后再观察多个规则块并行执行时可能产生的新问题，从而引入产生式规则的顺序锁定关系。为使下面的叙述明了起见，特规定如下表示法:

(1) 特殊记号$表示其后的字符串代表一个模式。

(2) $A 表示模式 A 与任意表达式匹配。

(3) $C 表示模式 C 与任意常数匹配。

(4) 设 E 是一个表达式，且已经和某个模式匹配上，则 $S(i) 表示在 E 的表达式(二元)树中先深编号(从 0 编起)为 i 的子表达式。其中:

(Ⅰ) E 本身是 $S(省略先深编号 0)。

(Ⅱ) 若 $S(i) 是 E 的一个子表达式，则它的左操作数是 $S(i+1)，右操作数是 $S(post(i))。函数 post(i) 的值，是在 E 的表达式树中按后根顺序求结点 i 的前导结点并取其先深编号而得的。如在(y↑z+x)*(e+f)中，$S(1) 是 y↑z+x; $S(2) 是 y↑z; $S(post(1)) 是 x; 见图 2

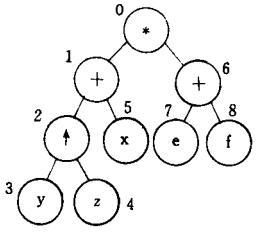


图 2　表达式树及其结点的先深编号

(5) 子表达式 $C<算子> $C 的值用 $V(i) 表示; i 是<算子>所在结点的先深编号。如 5*(7*(x+y))经下述两条产生式处理之后变成 35*(x+y):

$C*($C*$A)⇒($C*$C)*$A

$C*$C⇒$V

利用上述表示法，可以写出一个对数学表达式进行处理的产生式程序块:

(1) $C+$C⇒$V

(2) $C*$C⇒$V

(3) $C-$C⇒$V

(4) $C↑$C⇒$V

(5) $A+$C⇒$S(post(1))+$S(1)

(6) $A*$C⇒$S(post(1))*$S(1)

(7) $A↑0⇒1

(8) $A↑1⇒$S(1)

(9) 1*$A⇒$S(2)

(10) 0*$A⇒0

(11) 0+$A⇒$S(2)

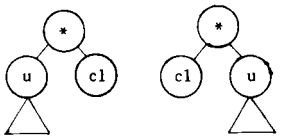
(12) $C*($C*$A)⇒($S(1)*$S(3))*$S(4)

这些规则的语义是明显的，如(6)是将图 3(a) 的表达式树变换成图 3(b)。象这样的规则还可以写出许多。但这里的主要目的是想说明基本问题，并不想追求产生式系统的完整性。问题是一个表达式 E 可能与几条规则的左部匹配，那么按什么顺序将规则用于处理 E 呢? 就是说:

(1) 当几个子表达式可以同时由一条或几条规则处理时，先处理哪个子表达式呢?

(2) 当几条规则可以同时用于处理同一个子表达式时，先用哪一条规则?

例如表达式 2+ 3 与(1)和(5)的左部都匹配。如果在用(5)之前先用(1)进行处理，则此产生式系统是正确的并且对表达式的变换将终止(收敛)；但是，如果每当能够使用(5)时总是先使用之，则在处理象 2+ 3 这样的表达式时将永不终止。



(a) (b)

图 3 经规则(6)变换的表达式树

对收敛性问题，上面是以具体的产生式系统为范例进行分析的；但分析结果对一般的产生式系统是同样适用的。因此，对一般的产生式系统，从单块的角度考虑其正确性，我们遵循如下执行顺序:

理好规则的排列顺序，使当两个规则都适用于同一个被处理对象时，先用排在前面的规则。

(2) 如果被处理对象 E 是一个复合对象，则按自左而右的顺序先处理所有子对象，再处理 E，见图 4。

(1) 整此处理过程可写成:

```
repeat
  if 被处理对象 E 是复合对象 then
    begin
      将所有可能的规则用于 E1;
      将所有可能的规则用于 E2;
      …
      将所有可能的规则用于 EK
    end;
    repeat
      将与 E 匹配的第 1 个规则用于 E
    Until 不再有适用的规则
until 可用于整个被处理对象 E 的规则已全部用尽;
```

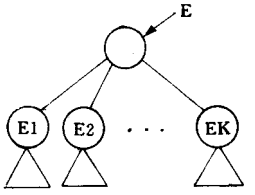


图 4 E 是一个复合对象

此过程一再重复，直到在某一次完全的扫描中，被处理对象没有任何改变(即没有任何规则可用于被处理对象的任何部位)为止。

要证明上述算法对任何产生式集合都终止，一般是不可能的，它对某些产生式集合是可终止的；对其它一些则不然。但它无疑提高了产生式系统执行正确性的可信度。

现在进而讨论多个规则块并行执行的情形。设系统中有规则 $L_1 \Rightarrow R_1$ 和 $L_2 \Rightarrow R_2$；顺序上 $L_1 \Rightarrow R_1$ 排列在前；若某对象 E 与 L_1 和 L_2 都匹配，则必须先执行 R_1 后执行 R_2 才正确。问题是，在产生式系统并行划分时，很可能将 L_1 和 L_2 分在后端的 T_i 和 T_j 的规则库中。执行时，前端机可能将 E 分别发送到 T_i 和 T_j 去并行匹配。如果 E 与 L_2 很快匹配成功，而 E 与 L_1 的匹配却迟得，则前端机可能早早就收到执行 R_2 的信号并错误地执行了 R_2。本文通过规则的顺序锁定关系解决这个问题。

定义 1. 设系统中有规则 $i_1: L_1 \Rightarrow R_1$，$i_2: L_2 \Rightarrow R_2$。其中 i_1 和 i_2 分别是二规则附带的标号。用序对 $\langle i_1, i_2 \rangle$ 表示: 如果两条规则都获得点火的机会，则执行右手部的顺序必须是先 R_1 后 R_2。我们说这两条规则是顺序锁定的。

定义 2. 设 $I = \{i_1, i_2, \cdots, i_n\}$ 是一个产生式系统中规则所附带的标号。按定义 1 构造的序对集 $\{(i, j) \mid i, j \in I\}$ 叫作该系统中规则的顺序锁定关系。

现在假设某系统的规则顺序锁定关系已经形成。对产生式系统的正确的顺序锁定应使 I 成为一个偏序集。在多个规则块并行执行时，当前端机收到信号，要求执行 $j: l \Rightarrow r$ 中的 r 时，应先检查一下: 若有顺序锁定(i, j)，且 $i: L \Rightarrow R$ 中的 R 尚未执行，则 r 的执行必须推迟到 R 之

后。

有了顺序锁定关系, 当把一个产生式系统 P 分成多个块时, 并行执行的收敛可能性不会比将P 作为单块串行执行的收敛可能性更差。

3　产生式系统的动态分析

对一般的产生式系统而言, 其顺序锁定关系的构造, 无论是手工做的还是用程序的, 都不敢说它的执行一定终止。利用动态分析的手段, 有可能发现隐蔽的死循环, 或为规则块的重新划分提供新的数据。

(1) 路径跟踪: 利用规则附带的标号, 可在产生式的运行过程中记下规则的执行路径。每当执行一条规则时, 按执行的先后顺序将(K, count(addr)) 插入一个运行图 G 中, 如果在此之前标号K 尚未在G 中出现的话。其中 count 记下该规则在这条路上执行的次数, addr 表示结点的地址。

随着系统的运行, 可能形成图 5 的 G。结点中的 k, $k_1, k_2, \cdots, k_1', k_2', \cdots$是已执行过的规则的标号。如果K 此前在 G 中出现过, 则 G 中将产生有向回路, 或无向回路。且该结点的 count(addr) > 1。通过对运行图 G 的分析, 能够发现产生式系统运行中是否出现 "循环"。因为 G 中存在有向回路的充要条件是 G 中有回退边(如(Km, k)); 而交叉边(如(k_1', k_3')) 的出现则使 G 中出现无向回路, 从而使回路的形态更加复杂。对象E 在 k_1 处经 k_1: $l_1 \Rightarrow r_1$ 处理时, 由于 E 自身瞬时状态的不同, 可能进入回路($k_1, k_2, k_3, \cdots, k, k_1$) 或($k_1, k_1', k_3, \cdots, k, k_1$) 或转入($k_1, k_2', k_3'$) 而终止。每当 G 中形成一条新的回路时, 回路中各结点的计数值都要重新设置。因此, 如果系统的执行在一个回路中长时间兜圈子, 则回路中各结点的 count 值相等, 而且这个值很大。这时可能进入了死循环, 可以考虑进行必要的人工干预。

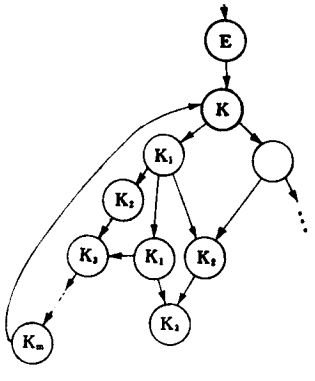


图 5　一个运行图 G

(2) 匹配成功率: 如何对产生式进行划分, 要考虑到粒度的大小, 各处理机间的负载平衡, 处理机之间通讯的多少等复杂因素, 很难划分得合理。因此在 PKBM 95 中采取 "静态并行划分; 再根据运行数据对子规则库进行改组"的方法。所谓匹配成功率就是运行时统计的数据之一, 是指在 T_i 中与规则左手部匹配上的对象元素数目与进入 T_i 的对象元素总数之比。如果匹配成功率低下, 就意味着有很多对象元素在处理机之间白白跑来跑去。影响匹配成功率的因素有: T_i 的规则库 Φ 太小; Φ 中各规则左手部之间相同或相似的条件元素太多; 等等。

(3) 匹配开销　指进入 T_i 的对象元素总数 n_i 与| Φ| 的乘积 O(n_i* | Φ| , 其中| Φ| 是 Φ 的大小。

(4) 匹配额外消耗: 指匹配成功率之倒数与| Φ| 的乘积。如果| Φ| 太大, 而且规则的并行划分又不太合理, 使对象元素 Φ 中未能获得较高的匹配成功率, 则额外销耗增大。

(5) 负载平衡度。可用 O(n_i* | Φ| /(n_j* | Φ| 来衡量。其中 n_i 和 n_j 分别是进入 T_i 和 T_j 的对象元素的总数。由于后端各 T_i 是相同的 Transputer CPU, 它们之间的负载平衡并不难

把握。只要| Φi| 与| Φj| 大体相同，每次进入 T_i 和 T_j 的对象元素总数 n_i 和 n_j 大体一样多，则 T_i 和 T_j 将接近平衡。但目前我们在前端机只模拟一台 Transputer CPU 的活动，能否在产生式系统运行时使前后端机达到基本平衡？这一点很难说清，因为前端机要执行对后端机的管理等复杂事务，其运行速度与 Transputer CPU 相差又比较大。因此只有实测才能得到更可靠的数据。说不定要在前端机模拟两台 Transputer CPU 更有利于前后端机的负载平衡。

上述五种数据不难在产生式系统运行时测定。它们对产生式系统的正确性和运行效率将起至关重要的作用。

4 结论

(1) 规则的顺序锁定能够使并行执行的收敛可能性不小于串行执行时收敛的可能性。

(2) 路径跟踪是一种检测隐蔽死循环的有应用价值的方法。

参 考 文 献

〔1〕 Steve kuo etal. The state of the art in parallel production systems. Journal of parallel and distributed computing 15, 1992 pp1- 26

〔2〕 郭福顺等. 一台并行知识库机的推理模型和通讯同步机制, 小型微型计算机系统. 1996 年第 10 期

〔3〕 C. L. Forgy, Rete: A Fast algorithm for the many pattern/ many object pettern match problem, Artificial intelligcnce 19, SEPT, 1982

〔4〕 Guo fushun et al. Research and design of parallel knowledge base machine- PKBM95, Journal of harbin institute of technology2, June 1996

RESEARCH ON PARALLEL EXECUTION AND CONVERGENCE PROBLEM OF PRODUCTION SYSTEMS

GUO Fushun SONG Zhen MENG Yao LIAO Minghong

(*Dept. of Computer Science of Harbin Institute of Technology, Harbin* 150001)

Abstract With the adoption of parallel technology to production systems, two new hard problems appeared: the compatibility problem and the convergence problem. Based on the parallel knowledge base machine PKBM95, in order to solve the convergence problem, this paper presents a method called sequential rule- locking. In addition, to find out potentialcycles and improve performance, we give a series of dynamic analysis and test methods, such as path- tracing, etc.

Key words Parallel knowledge base machine, Convergence problem, Sequential rule- locking